
CONQUER: Confusion Queried Online Bandit Learning

Daniel Barsky

Department of Electrical Engineering
Technion Institute of Technology
Haifa, Israel 3200003
danielbr@tx.technion.ac.il

Koby Crammer

Department of Electrical Engineering
Technion Institute of Technology
Haifa, Israel 3200003
koby@ee.technion.ac.il

Abstract

We present a new recommendation setting for picking out two items from a given set to be highlighted to a user, based on contextual input. These two items are presented to a user who chooses one of them, possibly stochastically, with a bias that favors the item with the higher value. We propose a second-order algorithm framework that members of it use *relative* upper-confidence bounds to trade off exploration and exploitation, and some explore via sampling. We analyze one algorithm in this framework in an adversarial setting with only mild assumption on the data, and prove a regret bound of $O(Q_T + \sqrt{TQ_T \log T} + \sqrt{T} \log T)$, where T is the number of rounds and Q_T is the cumulative approximation error of item values using a linear model. Experiments with product reviews from 33 domains show the advantage of our methods over algorithms designed for related settings, and that UCB based algorithms are inferior to greed or sampling based algorithms.

1 Introduction

Consider a book recommendation system triggered by users' actions. Given a book search performed by the user, one can generate heuristically about a dozen possible items to be presented, by using author identity, other books in a series, and so on; Yet, to be effective, only one or two results should be highlighted to a user. The question is, given a set of possible books, how to pick which two books should be presented to the user.

In this paper, we introduce confusion queried online bandit learning, that given a set of items, picks two items to be presented to the user. We assume the user will choose one of the two items stochastically based on the items' values (or rewards) for the user. Thus, if one item has clearly larger value than the other, the user will likely choose it. However, if values of both items are similar, the user will choose one at random. This relative feedback is also used in the dueling bandit setting [1, 2, 3, 4, 5].

We present CONQUER, a second-order online algorithm framework designed for this setting, and propose 7 algorithms within the framework. The framework assumes almost nothing about the data generation process. One algorithm uses *relative* upper confidence bounds, for which we show a regret bound that is $O(Q_T + \sqrt{T(Q_T + \log T)} \times \log T)$, where T is the number of rounds and Q_T is the approximation error of rewards using a linear model.

We evaluate the algorithms using product reviews from Amazon over 33 product domains. On each iteration, a few (between 5 and 20) possible items are chosen and presented to each algorithm, which picks two of them based on their reviews. The algorithm then receives a stochastic binary preference feedback, with a bias towards the item with the higher star rating, between 1 and 5 stars. Algorithms

were evaluated on a test set. The CONQUER algorithms show good performance on these tasks, generating better results than two baselines designed for a related setting.

Online learning with relative feedback is also useful in many applications where feedback is received from a human annotator. In such cases, producing a full annotation of each instance can be time-consuming and prone to mistakes. Producing a comparison between only two options can be much quicker and easier. One such setting is the sentence dependency parsing problem, where generating a full parse tree is much harder than deciding between two possible sources for a single word in a sentence. We also describe and analyze an algorithm for learning to parse sentences in natural language using light feedback regarding dependency parse trees [6]. In this setting, the algorithm can query for feedback on only a single edge, while it's evaluated on the entire parse tree.

2 Problem Setting

Online learning is performed in rounds or iterations. On iteration t , the algorithm receives a set of K ¹ items. The algorithm then chooses (indices of) two items, $m_t, n_t \in \{1, \dots, K\} \triangleq \mathcal{K}$, and receives a *stochastic* binary feedback, $y_t \in \{+1, -1\}$, indicating which of its two choices is better. A feedback of $y_t = +1$ indicates that item m_t has higher value or reward than item n_t , and vice-versa for $y_t = -1$. Additionally, the algorithm receives reward for the two items it picked.

We assume that the reward function is bounded, $|r(t, m)| \leq 1 \forall m \in \mathcal{K}$, and $t = 1, 2, \dots, T$. The reward value is not known or given to the algorithm, which is only evaluated by it. The algorithm then proceeds to the next round. Since we make almost no assumptions about the reward function, and it is never revealed to the algorithm directly, our algorithm operates in a fully adversarial setting - our only assumption is that the reward function can be approximated using a linear function, for if it's not the case, then there is no hope for any linear model based algorithm. The reward in our setting is deterministic, and all stochasticity in our model is assumed to be in the feedback process.

The feedback in our setting is assumed to be Bernoulli stochastic with a parameter that is proportional to the difference between the rewards of our two selected items, that is for $y \in \{\pm 1\}$,

$$\Pr(y_t = y) = \frac{1 + y \cdot \frac{1}{2}(r(t, m_t) - r(t, n_t))}{2} \quad (1)$$

Note that under this assumption, if $r(t, m_t)$ and $r(t, n_t)$ are close to each other, the feedback y_t will be either $+1$ or -1 with equal probabilities, whereas if the rewards are significantly different, the feedback will be biased towards the item with the higher reward.

Another interpretation for this setting is the feedback being (mostly) deterministic, but the *feedback provider* observing noisy rewards: instead of observing $r(t, m_t), r(t, n_t)$, she is shown $\hat{r}(t, m_t), \hat{r}(t, n_t)$, which are given as follows:

$$\hat{r}(t, m) = \begin{cases} +1 & w.p. \frac{1+r(t, m)}{2} \\ -1 & w.p. \frac{1+r(t, m)}{2} \end{cases}$$

The feedback provider returns $y_t = +1$ if $\hat{r}(t, m_t) > \hat{r}(t, n_t)$, $y_t = -1$ if $\hat{r}(t, m_t) < \hat{r}(t, n_t)$, and ties are broken arbitrarily with equal probabilities. Regret is now calculated using *expected* noisy rewards. Proof of the equivalence appears in App. A.3 in the supp. material.

Let r_t denote the instantaneous regret at round t , which we define as the difference between the reward of the best item, and the reward the algorithm received:

$$r_t = \max_{m \in \mathcal{K}} r(s, m) - \max\{r(s, m_s), r(s, n_s)\} . \quad (2)$$

The goal of the algorithm is to minimize its cumulative regret, $R_t = \sum_{s=1}^t r_s$. We denote the optimal item by $m_t^* = \arg \max_{m \in \mathcal{K}} r(t, m)$. The regret is with respect to the better of the two items since we assume a user is looking for a single item, as long it has very high value - one very high-valued item and one very low-valued item are a better combination than two mid-value items, even though the latter might have higher average value.

¹The algorithm and analysis can be extended to the case where there are a different number of items on each round, that is $K = K(t)$. We use a fixed value K for simplicity.

We focus on linear models. For each item $\mathbf{x}_{t,m}$, let $\Phi(\mathbf{x}_{t,m}) = \Phi(\mathbf{x}_t, m) \in \mathbb{R}^D$ denote a feature vector representing the m th item in the set \mathbf{x}_t . We assume that the feature vector is bounded and of unit Euclidean norm, that is, $\|\Phi(\mathbf{x}_t, m)\| = 1$, for all $t = 1, 2, \dots$, $m \in \mathcal{K}$. We wish to approximate the reward function using a linear function, for some vector $\mathbf{u} \in \mathbb{R}^D$, $\mathbf{u}^\top \Phi(\mathbf{x}_t, m)$. We assume the dot product between the vector \mathbf{u} and the feature vector of *any* incoming instance is bounded by 1 in absolute value, meaning $|\mathbf{u}^\top \Phi(\mathbf{x}_t, m)| \leq 1$ for all $m \in \mathcal{K}, t$.

3 Related Work

We compare our framework to similar relevant methods in terms of regret formulation, feedback, assumptions and more in Table 2 in the supp. material. In addition, Table 1 in the supp. material shows our method side-by-side with similar contextual bandits methods with partial feedback. In this section, we outline previous works similar or relevant to ours.

The dueling feedback model was initially explored in the bandits setting, where several explore-then-exploit solutions were proposed [1, 2]. The regret in these settings is more general than ours, since they do not assume any reward for a specific arm - only comparison results between two arms, their regret being proportional to the probability that one arm will beat another in such a comparison. If we assume that each arm has a reward associated with it, and that the probability of an arm defeating another arm is proportional to the difference in rewards between the two arms, then the *weak regret* defined there is equivalent to our regret formulation. The *strong regret* used there is stronger than ours. An algorithm for interactively training information retrieval was proposed for the online learning as well [3]. Another similar setting is the online learning with preference feedback [4, 5]. A comprehensive survey on bandit learning was recently published [7]. All this line differs from our, as we consider additional side information, or context.

There has been a considerable amount of work on contextual online prediction with light feedback. The Banditron algorithm [8] receives binary true-false feedback for a single prediction, and maintains a $O\left(T^{\frac{2}{3}}\right)$ regret bound. The Banditron algorithm was extended in few ways [9, 10]. Confidit [11] operates in a similar setting to Banditron, but assumes a specific stochastic label generating mechanism, and achieves $O\left(\sqrt{T} \log T\right)$ regret. Our work builds on this, but has few major differences: First, Confidit uses binary (true-false) feedback on a single prediction, while we use binary relative (dueling) feedback. Second, they assume a specific stochastic label generation process, while we do not. This assumption is expressed in the regret, where Confidit assumes stochastic regret, whereas we use a general and deterministic regret formulation. Third, their analysis is on the expected zero-one reward, while ours is a high-probability bound over any bounded reward. Fourth, their feedback is deterministic, while ours is allowed to be noisy (or stochastic). Finally, the multiclass categorization problem they consider is a special case of our setting, the reward of a single item being 1, and the reward of the rest being 0. We stress that most if not all previous works on UCB construct *absolute* confidence bounds based on the estimated quantities, while we construct both absolute and *relative* confidence bounds between two elements (see (3)).

A similar linear model to ours was used in a setting with true-false-don't know feedback (the KWIK setting) on a single prediction [12]. This work was extended and showed some improved results [13]. Newtron [14] is a second order descent method for the online multiclass bandit setting, for which there is a bound on the *log-loss*, unlike the direct regret formulation we used. A similar solution was used for binary classification from single and multiple teachers [15], but with binary (true-false) feedback. A similar algorithm was also used in the ordered ranking problem, with feedback indicating the intersection between the predicted and optimal ordered results [16]. Their regret formulation is also general, yet it takes into account the number of labels output (which is constant in our case), and the order in which the labels were output, which we do not care about. In addition, their algorithm requires several parameters (as opposed to our single parameter), and makes several assumptions which we do not.

Using confidence bounds to trade off between exploration and exploitation has been proposed [17] and later extended [18]. However, their work uses a confidence bound in a single prediction to evaluate the accuracy of a prediction, while we use similar methods to evaluate the possible confusion between two possible predictions. Multiclass online learning with bandits feedback can be consid-

ered a bandit problem with side information. An epoch-greedy algorithm for contextual multi-armed bandits has been introduced [19], with a regret bound of $O(\sqrt{T} \log T)$.

4 Confusion Queried Online Bandit Learning

We now describe an online, second order algorithm framework for the recommendation setting described above. This framework will define a group of algorithms, which differ only in their method of selecting the two items to be displayed m_t, n_t , and are identical otherwise. Every algorithm in our framework maintains a linear model, $\mathbf{w}_t \in \mathbb{R}^D$, which it uses to estimate the reward of each item. In addition, our algorithms maintain a positive definite (PD) matrix, $A_t \in \mathbb{R}^{D \times D}$, used sometimes to estimate the confidence in the score of an item or the confusion between two possible items from a given set of items \mathbf{x}_t . We update \mathbf{w}_t and A_t using the standard second-order update rule, using the vector $\mathbf{z}_t = \frac{1}{2}y_t \cdot (\Phi(\mathbf{x}_t, m_t) - \Phi(\mathbf{x}_t, n_t))$ as the update vector, where $y_t \in \{\pm 1\}$ is the feedback received by the algorithm. Let us define the following Mahalanobis distance between two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^D$, $d_t(\mathbf{a}, \mathbf{b}) = \frac{1}{2}(\mathbf{a} - \mathbf{b})^\top A_t (\mathbf{a} - \mathbf{b})$, and let $\tilde{\mathbf{w}}_{t-1} \in \mathbb{R}^D$ denote an orthogonal projection of \mathbf{w}_{t-1} w.r.t. the above-mentioned distance that satisfies $|\tilde{\mathbf{w}}_{t-1}^\top \Phi(\mathbf{x}_t, m)| \leq 1 \forall m \in \mathcal{K}$. This projection stage is required for our regret analysis. Let $\hat{\Delta}_{t,m} = \tilde{\mathbf{w}}_{t-1}^\top \Phi(\mathbf{x}_t, m)$, denote the score of item m under our model. Since the reward at round t is the maximum of the rewards of our two predictions, we are OK if we output $m_t^* = \arg \max_{m \in \{1, \dots, K\}} \Delta_{t,m}$ as one of our predictions.

Let $\epsilon_{t,m}$ denote the amount of confidence we have in the score of item m according to our model, $\epsilon_{t,m}^2 = \eta_t \times \Phi(\mathbf{x}_t, m)^\top A_{t-1}^{-1} \Phi(\mathbf{x}_t, m)$. A higher value of $\epsilon_{t,m}$ implies less confidence in the value of $\hat{\Delta}_{t,m}$. Let $\epsilon_{t,m,n}$ denote the amount of our confusion between items $m, n \in \{1..K\}$ in the set \mathbf{x}_t :

$$\epsilon_{t,m,n}^2 = \eta_t \times (\Phi(\mathbf{x}_t, m) - \Phi(\mathbf{x}_t, n))^\top A_{t-1}^{-1} (\Phi(\mathbf{x}_t, m) - \Phi(\mathbf{x}_t, n)) \quad (3)$$

$$\eta_t = d_0(\mathbf{u}, \mathbf{0}) + 2 \sum_{s=1}^t q_s + 2 \sum_{s=1}^{t-1} \mathbf{z}_s^\top A_{s-1}^{-1} \mathbf{z}_s + 36 \ln \frac{t+4}{\delta} \quad (4)$$

Here, q_s is the approximation error of the reward function using a linear model at time t , and δ is a parameter of the algorithm, indicating with what probability the cumulative regret is bounded by the regret bound. This form of η_t is required only for the analysis, and is not practical. In practice, we set $\eta_t = \eta$ to be a constant value.

On iteration t , each of our algorithms receives a set \mathbf{x}_t , and starts by projecting its current model vector, \mathbf{w}_{t-1} , onto the set mentioned above. The algorithm then pick two items to be displayed. Once the two items m_t, n_t are selected by the algorithm, it receives the relative stochastic binary feedback $y_t \in \{\pm 1\}$ and performs a standard second order update using the difference vector \mathbf{z}_t . We call the algorithm framework *CONQUER* for CONFusion QUERied online bandit learning. Below we also use CNQR for a shorter abbreviation. Algorithm 1 describes our framework formally. In addition, Table 1 shows the method of selecting n_t in 3 algorithms we propose within our framework.

We will provide regret analysis for the CNQR-GNC algorithm. In this algorithm, the first item m_t is chosen as the best item according to the model at time t , $m_t = \arg \max_{m \in \mathcal{K}} \hat{\Delta}_{t,m}$, and the second item is the "optimistically best" item (that's different from m_t). In other words, we define relative upper-confidence scores, $\beta(n) = \hat{\Delta}_{t,n} - \hat{\Delta}_{t,m_t} + \epsilon_{t,n,m_t}$.

Algorithm 1 CONQUER - Framework Outline

Input: $\delta \in (0, 1)$ (used in (3) and (4))

1: Initialize $\mathbf{w}_0 = \mathbf{0} \in \mathbb{R}^D$, $A_0 = \mathbf{I}_{D \times D}$

2: **for** $t = 1$ to T **do**

3: Receive set of items $\mathbf{x}_t \in \mathcal{X}^K$

4: Project $\tilde{\mathbf{w}}_{t-1} = \arg \min d_t(\mathbf{w}, \mathbf{w}_{t-1})$
 Subject to $|\mathbf{w}^\top \Phi(\mathbf{x}_t, m)| \leq 1 \forall m \in \mathcal{K}$

5: Set $m_t = \arg \max_{m \in \mathcal{K}} \hat{\Delta}_{t,m}$

6: Set n_t according to Table 1

7: Output m_t, n_t

8: Receive feedback $y_t \in \{\pm 1\}$

9: Set $\mathbf{z}_t = \frac{1}{2}y_t \cdot (\Phi(\mathbf{x}_t, m_t) - \Phi(\mathbf{x}_t, n_t))$

10: Update $A_t = A_{t-1} + \mathbf{z}_t \mathbf{z}_t^\top$

11: Update $\mathbf{w}_t = A_t^{-1} (A_{t-1} \tilde{\mathbf{w}}_{t-1} + \mathbf{z}_t)$

12: **end for**

Output: $\tilde{\mathbf{w}}_T$

Next, we output the best item in terms of $\beta(n)$ that is different from our first choice, $n_t = \arg \max_{n \in \mathcal{K} / \{m_t\}} \beta(n)$. We note that it is a mixture of first-order (as m_t is just the best according to the model) and upper confidence bound (as n_t is chosen using a confidence bound) strategies. Additionally, these confidence bounds are *relative* to the best action m_t , and not absolute as done in general. Finally, we allow CNQR-GNC *not* to pick a second item, if it is certain that the first item picked is the best one, even in face of uncertainty, that is if $\beta(n_t) < 0$. Conceptually, if the difference between the reward of the two item is higher than the confidence interval, then we are sure the first item is better, and the second one can be ignored. In this case no update will be performed, as there is no relative feedback.

Computing the Projection: The algorithm requires computing an orthogonal projection problem, which means minimizing a quadratic function subject to (multiple) linear constraints, $\tilde{\mathbf{w}}_{t-1} = \arg \min_{\{\mathbf{w} \mid \mathbf{w}^\top \Phi(\mathbf{x}_t, m) \leq 1 \ \forall m \in \mathcal{K}\}} d_t(\mathbf{w}, \mathbf{w}_{t-1})$. This problem is convex, yet we could not find a closed form solution. We propose to solve it using a sequential algorithm which projects on a single constraint [20]. A detailed description of this method can be found in App. A.4.

Time and Space: We conclude this section by noting that each step of the algorithm requires $O(D^2 + DK)$, where $O(D^2)$ is needed to compute the inverse (and the projection) and the update of the matrix A_t , and $O(DK)$ to compute the output and update the parameters \mathbf{w}_t . Note also the algorithm can also be run in dual variables (i.e., in a RKHS). This has a twofold implication: (a) The resulting reward model (2) can be made highly non-linear in the features, and (b) the running time per round can be made quadratic in the number of rounds so far.

In the experiments reported below, we used a version of the algorithm that maintains and manipulates a diagonal matrix \mathbf{A} instead of a full one. All the steps of the algorithm remain the same, except the update $A_t = A_{t-1} + \mathbf{z}_t \mathbf{z}_t^\top$ of line 17, that is replaced with updating only the diagonal elements, $(A_t)_{r,r} = (A_{t-1})_{r,r} + (\mathbf{z}_t)_r^2$. The running time is reduced now to $O(KD)$ (the projection depends also on the number of iterations). The space needed is now D for both \mathbf{w} and \mathbf{A} .

Name	n_t Selection
<i>Top Two Greedy</i> (CNQR-TTG)	Second best in terms of score $\arg \max_{n \in \mathcal{K} / \{m_t\}} \hat{\Delta}_{t,n}$
<i>Greedy + Random</i> (CNQR-GNR)	Random item from the remaining set Random $\{\mathcal{K} / \{m_t\}\}$
<i>Greedy + UCB</i> (CNQR-GNU)	Best in terms of UCB $\arg \max_{n \in \mathcal{K} / \{m_t\}} \hat{\Delta}_{t,n} + \epsilon_{t,n}$
<i>Greedy + Confusion</i> (CNQR-GNC)	Best in terms of <i>relative</i> UCB $\arg \max_{n \in \mathcal{K} / \{m_t\}} \hat{\Delta}_{t,n} + \epsilon_{t,m_t,n}$ (doesn't query or update if $\hat{\Delta}_{t,m_t} - \hat{\Delta}_{t,n_t} > \epsilon_{t,m_t,n_t}$)

Figure 1: Algorithms within the CONQUER framework

5 Regret Analysis for CNQR-GNC

We have no assumption on the data generation process, that is, how items \mathbf{x}_t were generated, and how the reward function $r(t, m)$ is defined. Yet, we are approximating the latter with a linear function of the former (or its features). If the reward function is far from being linear, then there is no hope for the algorithm to work well. We thus quantify the approximation error, and define $q_t = \max_{m \in \mathcal{K}} |r(t, m) - \mathbf{u}^\top \Phi(\mathbf{x}_t, m)|$, and let $Q_t = \sum_{s=1}^t q_s$ denote the cumulative approximation error. In the analysis below, we compare the performance of our algorithm to the performance of any linear model \mathbf{u} . We therefore denote by $\Delta_{t,m} = \mathbf{u}^\top \Phi(\mathbf{x}_t, m)$, the approximate reward of item $\mathbf{x}_{t,m}$. Thus, the label maximizing the approximate reward is, $m_t^* = \arg \max_{m \in \mathcal{K}} \mathbf{u}^\top \Phi(\mathbf{x}_t, m)$, with associated approximate reward $\Delta_{t,m_t}^* = \mathbf{u}^\top \Phi(\mathbf{x}_t, m_t^*)$.

We now compute a bound on the regret R_t defined in (2). The regret is a sum of two terms: approximation error (due to approximating the reward function with a linear model) and estimation regret (due to stochastic feedback and online learning). The approximation error measures how well our (linear) model is, and is not algorithm dependent.

First, we bound the instantaneous regret by a sum of the approximation error q_t and the confusion coefficient for that round, under the assumption that $\left| (\hat{\Delta}_{t,n_t} - \hat{\Delta}_{t,m_t}) - (\Delta_{t,n_t} - \Delta_{t,m_t}) \right|$ is bounded by the confusion coefficient, for any t, m_t and n_t (Lemma 2). Then, we show that

this assumption holds with high probability (Lemma 3, Lemma 4, Lemma 5; all given in the App. A.1 in the supplementary material). Finally, we bound in Theorem 1 the cumulative regret by $O(Q_T + \sqrt{T + Q_T} \log T)$. The last term is small if the approximation error Q_T is small. Our proof builds on results developed of [11]. The main result of this section is Theorem 1, which its proof appears in App. A.1 and App. A.2 in the supp. material.

Theorem 1. *In the setting described so far, the cumulative regret R_t of CNQR-GNC satisfy $R_t = 2Q_T + \sqrt{2T} \left(\sqrt{((2Q_T + A)B)} + B \right)$, with probability at least $1 - \delta$ uniformly over the time horizon T , where $A = d_0(\mathbf{u}, \mathbf{0}) + 36 \ln \frac{T+4}{\delta}$ and $B = 2DK \ln \left(1 + \frac{T}{dK}\right)$.*

6 Experimental Study

We evaluated our algorithms using the following recommendation setting on reviews from Amazon. We used 341,400 reviews on products from 33 domains². The reviews were preprocessed by converting upper-case text to lower-case, replacing common non-word patterns (such as common emoticons, 3 dots, links) with a unique mark, removing HTML tags, and expanding abbreviations. We extracted bi-gram features, with 6,255,811 features per item. Each review also comes with a rating, between 1 and 5 stars, which was used as its (normalized) reward.

On each iteration, each algorithm is given a set of K items from a specific domain, each represented by a review written on that item. The algorithm then picks two items m_t, n_t based on these reviews, and receives a stochastic bit $y \in \{\pm 1\}$ with probability proportional to the difference in the number of stars each of the items received. If the number of stars both reviews got are close, the bit will be $y = +1$ with probability ≈ 0.5 . Where the difference is maximal (4), the bias is $(1 + 0.25 \times (5 - 1))/2 = 1$. This process simulates the case where the two items are shown to a user, who clicks on one item based on its relative value or reward. Even though full feedback is available here, relative feedback simulates better the process of choosing one out of several suggested products occurring in practice.

The error at time t was defined as the difference between the maximal rating in the reviews received at time t and the rating of the first review selected by the algorithm (m_t), divided by 4. The highest value is 1, and choosing a random review as the best yields an average error of 0.5. We experimented with sets of size $K = 5, 10, 15, 20$. In total, we have $33 \text{ (domains)} \times 4 \text{ (values of } K) = 132 \text{ experiments}$. Each trial was performed 10 times, and the results were averaged. The error bars appearing in the figures indicate the 95 percentile.

²android, arts, automotive, baby products, beauty, books, camera, cell-phones, clothing, computers, electronics, food, gardening & pets, health, home, industrial, jewellery, kindle, kitchen, magazines, movies & TV, MP3, music, musical instruments, office, patio, shoes, software, sports, toys, video games, videos, watches

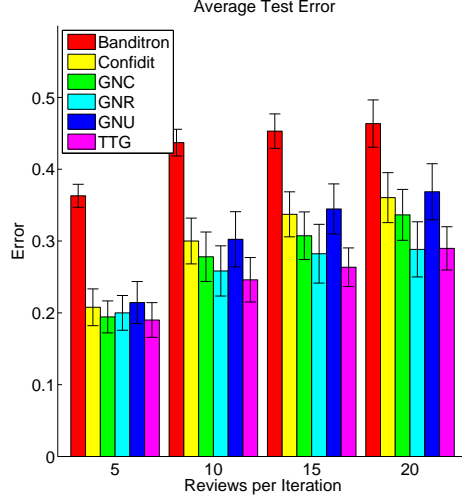


Figure 2: Average error over all Amazon domains

One-vs-One comparison – overall

TTG	0	86	113	129	132	132
GNR	46	0	94	116	125	130
GNC	19	38	0	118	125	130
Confidit	3	16	14	0	75	130
GNU	0	7	7	57	0	128
Banditron	0	2	2	2	4	0
	TTG	GNR	GNC	Confidit	GNU	Banditron

Figure 3: One-vs-one competition of all algorithms over all experiments

The reviews in each domain were divided into three sets: 15% was used as a development set to calibrate the parameter of each algorithm, 10% was used as a test set, and the remaining 75% were used for training. All algorithms had their parameters tuned on the development set, over a grid of fixed size, separately per domain.

Once parameters were tuned, we executed all algorithms on the training set (single iteration) and evaluated the resulting model on the test set. We evaluated a total of six algorithms. In addition to the CONQUER algorithms we outlined in Table 1, we also evaluated two multiclass contextual bandits algorithms - Confidit [11] and Banditron [8] - each picks a single item, and receives a binary feedback stating whether the picked item’s rating is maximal.

All algorithms that maintain second order information - the matrix A_t - were executed with a diagonal matrix. Confidit and the CONQUER algorithms require a scalar quantity η_t to be computed based of unknown quantities, such as the vector \mathbf{u} . Instead, we follow previous practice [11], and set $\eta_t = \eta$ to a fixed value tuned on the development set. Finally, since even now the projection is very time-consuming, and since it is required only for the sake of analysis, and was not improving performance in practice on small scale problems, it was not performed in the experiments below, where the data dimension is high.

Fig. 2 shows average test error over all domains. The four blocks correspond to the number of possible items per iteration K , and the eight bars per block correspond to the eight algorithms run. Fig. 3 shows a one-vs-one competition of all algorithms one against the other in all trials - a value of k in row i , column j indicates that algorithm i did better than algorithm j in k trials. These results are consistent with each other, meaning that algorithms that do well on average also beat less successful algorithms in a majority of the trials. Finally, Fig. 4 shows the test error for Automotive, Books, and Kindle domains. These results show a similar trend to the average results. Additional results are in App. A.7 in the supp. material.

Clearly, Banditron performs worst with error greater than 0.3, CNQR-TTG is the clear winner in both average error and trial wins, with CNQR-GNR in close second place. CNQR-GNC is third, and Confidit and CNQR-GNU follow. We also experimented with algorithms employing a non-relative UCB policy to select m_t , but these yielded poor results. Few comments are in order. First, *relative* UCB indeed outperforms absolute UCB in the setting described above, providing justification for this choice, and our choice of analysis. Second, it seems that greed or random exploration are better-suited to this setting than a UCB policy, both in the absolute and relative variants. Third, the performance of all algorithms relying on UCB policies deteriorates much faster as the number of items per round increases compared to non-UCB algorithms - we hypothesize that this is because as the number of items per round increases, the chance of choosing two items with identical reward grows, and for these pairs of items the UCB factor is more significant, whereas algorithms that choose based on score alone don’t suffer as much from the added confusion. Fourth, we evaluated additional algorithmic variants where the first choice of the algorithm is based on UCB, and the second is either second best UCB, or a random choice. Both these variants performed poorly compared to the variants we tried, and thus are omitted.

Finally, it is evident that avoiding making a query in some cases doesn’t seem to hurt performance significantly. However, we must point out that CNQR-GNC avoided making queries on a maximum of 8.27% of the rounds per domain, with common values ranging around 0.5% – 1% (the full statistics of optional querying are in Table 3 in the supp. material). This is due to the relatively small number of rounds per domain, and we expect this number to increase if multiple iterations over the training data are performed or the trial is run on a larger dataset.

7 Light feedback in Dependency Parsing

A setting related to ours is learning dependency parse trees with light feedback [6], where items are sentences and are associated with full parse trees. These trees are often generated by human annotators, in a process that is complex, slow and prone to mistakes, as for each sentence a full correct feedback is required. We focus on partial feedback, where given a sentence, the algorithm outputs a *complete* parse tree, but can request feedback on only a single word in the sentence, and feedback is only required about the relative correctness of two alternative edges related to that word.

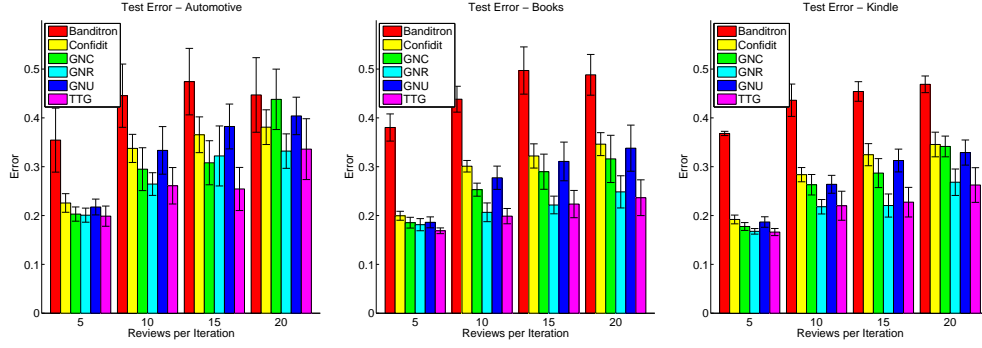


Figure 4: Average test error over Automotive, Books and Kindle domains.

More formally, an item received at time t is a sentence \mathbf{x}_t of length $s_t \leq S$ in some natural language. The goal of the algorithm is to generate a dependency parse tree \mathbf{m}_t between the words in the sentence. We assume that features about a tree decompose to features about edges, and specifically the syntactic relation between the i th word and the j th word can be captured in a feature vector $\phi(\mathbf{x}_t, i, j) \in \mathbb{R}^D$, and the score of an edge between words i and j is given by $\tilde{\mathbf{w}}_{t-1}^\top \phi(\mathbf{x}_t, i, j)$. Given a possible dependency parsing \mathbf{m} over \mathbf{x}_t , we construct an aggregated feature vector, $\Phi(\mathbf{x}_t, \mathbf{m}) = \frac{1}{s_t-1} \sum_{(i,j) \in \mathbf{m}} \phi(\mathbf{x}_t, i, j)$. The $\frac{1}{s_t-1}$ factor is required to ensure that $\Phi(\mathbf{x}_t, \mathbf{m})$ is of unit norm. Given a model weight vector $\tilde{\mathbf{w}}_{t-1}$, the predicted parse tree is computed by $\mathbf{m}_t = \arg \max_{\mathbf{m} \in \mathcal{K}_t} \tilde{\mathbf{w}}_{t-1}^\top \Phi(\mathbf{x}_t, \mathbf{m})$.

The confusion (or dueling) feedback in this setting is realized in the feedback given to the algorithm. Given some parse tree, e.g. the parse tree \mathbf{m}_t maximizing the score, the algorithm selects some word index $i \in \{1 \dots s_t\}$ and chooses an alternative source for it. The feedback it asks for and receives can be summarized in the question: “Which of the j th and k th words of the sentence are a better source for the i th word?”. For example in the sentence “*I saw the dog with the telescope*” the algorithm may ask “*Was the dog seen by a telescope, or was the dog seen carrying a telescope?*”. While in this case, both answers may be valid both syntactically and semantically, it is not the case when we replace the word *telescope* with the word *bone*. The algorithm then uses this feedback to update its model. This type of focused relative feedback makes the work of a human annotator much easier - instead of parsing an entire sentence, she is only required to answer to a simple yes/no question.

Our solution to this setting, CNQR-DP (CONQUER Dependency Parsing), is outlined in Algorithm 2, and Theorem 7 shows for it a regret bound of $O\left(S \cdot \sqrt{T} \log T\right)$ with respect to the edge-accuracy evaluation measure, often used in dependency parsing. Due to lack of space, both algorithm and analysis appear in App. A.4 in the supp. material. A similar previous solution [6] picks the best two possible alternatives greedily, while we pick one greedily and the other optimistically. Finally, we prove a regret bound, while they do not.

8 Conclusions and Future Work

We have introduced confusion queried online bandit learning, which shares properties with both contextual bandits and dueling bandits. We described a new algorithm framework for this task, suggested few algorithms in it, and analyzed the regret for one of them under very mild assumptions. We showed that the proposed online second order algorithms are efficient. Extensive experiments we performed with a recommendation system based on reviews showed the usefulness of the setting and our framework. Two main insights are: (1) in this settings it seems that greedy or random exploration outperform UCB based exploration (2) Relative UCB (as we analyzed) outperform absolute UCB. Thus, exploration should be tuned appropriately when a relative preference choice is performed.

Future work might include bounding the number of queries the algorithm makes. We also plan to explore similar frameworks where more than two labels can be presented for feedback, which would

pick the most relevant subset from the set selected by the algorithm. Last, but not least, we plan to experiment with our dependency parsing solution, and apply our methods to other domains.

References

- [1] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.
- [2] Yisong Yue and Thorsten Joachims. Beat the mean bandit. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 241–248, 2011.
- [3] Yisong Yue and Thorsten Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1201–1208. ACM, 2009.
- [4] Pannagadatta K Shivaswamy and Thorsten Joachims. Online learning with preference feedback. *arXiv preprint arXiv:1111.0712*, 2011.
- [5] Pannaga Shivaswamy and Thorsten Joachims. Online structured prediction via coactive learning. *arXiv preprint arXiv:1205.4213*, 2012.
- [6] A. Mejer and K. Crammer. Confidence in structured-prediction using confidence-weighted models. In *EMNLP*, 2010.
- [7] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 2012.
- [8] Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multi-class prediction. In *ICML*, pages 440–447. ACM, 2008.
- [9] R. Wang, S. and Jin and H. Valizadegan. A potential-based framework for online multi-class learning with partial feedback. In *ICAIS*, 2010.
- [10] H. Valizadegan, R. Jin, and S. Wang. Learning to trade off between exploration and exploitation in multiclass bandit prediction. In *SIGKDD*, 2011.
- [11] Koby Crammer and Claudio Gentile. Multiclass classification with bandit feedback using adaptive regularization. *Machine Learning*, 90(3):347–383, 2013.
- [12] T.J Walsh, I. Szita, C. Diuk, and M. Littman. Exploring compact reinforcement-learning representations with linear regression. In *UAI*, 2009.
- [13] H. Ngo, M. Luciw, N.A. Vien, and J. Schmidhuber. Upper confidence weighted learning for efficient exploration in multiclass prediction with binary feedback. In *IJCAI*, 2013.
- [14] Elad Hazan and Satyen Kale. Newtron: an efficient bandit algorithm for online multiclass prediction. In *NIPS*, pages 891–899, 2011.
- [15] Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Selective sampling and active learning from single and multiple teachers. *JMLR*, 13, 2012.
- [16] Claudio Gentile and Francesco Orabona. On multilabel classification and ranking with partial feedback. In *NIPS*, pages 1160–1168, 2012.
- [17] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.
- [18] Varsha Dani, Thomas P Hayes, and Sham M Kakade. Stochastic linear optimization under bandit feedback. In *COLT*, pages 355–366, 2008.
- [19] John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in neural information processing systems*, 20:1096–1103, 2007.
- [20] Y. Censor and S.A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, New York, NY, USA, 1997.
- [21] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *Siam Journal of Computation*, 34(3):640–668, 2005.
- [22] Jack Edmonds. *Optimum branchings*. National Bureau of standards, 1968.

A Supplementary Material

A.1 Lemmas

Lemma 2. *If at time t , the algorithm maintains $\left|(\hat{\Delta}_{t,n} - \hat{\Delta}_{t,m}) - (\Delta_{t,n} - \Delta_{t,m})\right| \leq \epsilon_{t,n,m}$ for all $m, n \in \mathcal{K}$, then $r_t = \max_{m \in \mathcal{K}} r(t, m) - \max\{r(t, m_t), r(t, n_t)\} \leq 2q_t + 2\epsilon_t$, where $\epsilon_t^2 = 2\mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t \times \eta_t$ for all $t = 1, 2, \dots, T$.*

Proof. Notice that from the description of the algorithm, $\hat{\Delta}_{t,n} - \hat{\Delta}_{t,m_t} \leq 0$ for any $n \in \mathcal{K}$. In addition, notice that $\epsilon_{t,m,n} = \epsilon_{t,n,m}$ for any $m, n \in \mathcal{K}$. From the algorithm description, we have,

$$\begin{aligned} r_t &= \max_{m \in \mathcal{K}} r(t, m) - \max\{r(t, m_t), r(t, n_t)\} \leq \Delta_t^* - \max\{\Delta_{t,m_t}, \Delta_{t,n_t}\} + 2q_t \\ &\leq \Delta_t^* - \Delta_{t,m_t} + 2q_t \leq \hat{\Delta}_{t,m_t^*} - \hat{\Delta}_{t,m_t} + 2\epsilon_{t,m_t^*,m_t} + 2q_t \\ &\leq \hat{\Delta}_{t,n_t} - \hat{\Delta}_{t,m_t} + 2\epsilon_{t,n_t,m_t} + 2q_t \leq 2\epsilon_t + 2q_t. \end{aligned}$$

Note that if $\beta(n_t) < 0$, the linear approximation term is bounded by a negative number, and since it is non-negative by definition, this means that the linear term is zero, and the regret is composed of the approximation error term alone. For this case, $\epsilon_t = 0$, and the inequality holds. \square

Lemma 3. *With the notation introduced so far, the following inequality holds for any t and any $\mathbf{v} \in \mathbb{R}^D$, where we have $h_s = \mathbf{z}_s^\top A_s^{-1} \mathbf{z}_s$*

$$\begin{aligned} \left(\tilde{\mathbf{w}}_{t-1}^\top \mathbf{v} - \mathbf{u}^\top \mathbf{v}\right)^2 &= 2\mathbf{v}^\top A_{t-1}^{-1} \mathbf{v} \cdot \\ &\quad \left(d_0(\mathbf{u}, \mathbf{0}) + 2 \sum_{s=1}^{t-1} h_s - \frac{1}{2} \sum_{s=1}^{t-1} \left(1 - \tilde{\mathbf{w}}_{s-1}^\top \mathbf{z}_s\right)^2 + \frac{1}{2} \sum_{s=1}^{t-1} \left(1 - \mathbf{u}^\top \mathbf{z}_s\right)^2\right) \end{aligned}$$

Proof. Let s be any round between round 1 and round $t-1$. Using the Cauchy-Schwarz theorem for dual norms, we have,

$$\left(\tilde{\mathbf{w}}_{s-1}^\top \mathbf{v} - \mathbf{u}^\top \mathbf{v}\right)^2 = |\langle \tilde{\mathbf{w}}_{s-1} - \mathbf{u}, \mathbf{v} \rangle|^2 \leq 2 \cdot \mathbf{v}^\top A_{s-1}^{-1} \mathbf{v} \cdot d_{s-1}(\tilde{\mathbf{w}}_{s-1}, \mathbf{u}). \quad (5)$$

Next, similarly to [11], we have,

$$d_s(\tilde{\mathbf{w}}_{s-1}, \mathbf{w}_s) = \frac{1}{2} \left(1 - \tilde{\mathbf{w}}_{s-1}^\top \mathbf{z}_s\right)^2 \cdot \mathbf{z}_s^\top A_s^{-1} \mathbf{z}_s \leq 2 \cdot \mathbf{z}_s^\top A_s^{-1} \mathbf{z}_s.$$

Again, as shown in [11], observe that

$$d_{s-1}(\mathbf{u}, \tilde{\mathbf{w}}_{s-1}) - d_s(\mathbf{u}, \mathbf{w}_s) + d_s(\tilde{\mathbf{w}}_{s-1}, \mathbf{w}_s) = \frac{1}{2} \left(1 - \tilde{\mathbf{w}}_{s-1}^\top \mathbf{z}_s\right)^2 - \frac{1}{2} \left(1 - \mathbf{u}^\top \mathbf{z}_s\right)^2$$

Now, observe that since $|\mathbf{u}^\top \mathbf{z}_t| \leq 1$ holds for any $t = 1, 2, \dots, T$, we can use the standard Bregman projection inequality stating that $d_s(\mathbf{u}, \tilde{\mathbf{w}}_s) \leq d_s(\mathbf{u}, \mathbf{w}_s)$. Plugging this into the above, and summing over s , we get:

$$\frac{1}{2} \sum_{s=1}^{t-1} \left(1 - \tilde{\mathbf{w}}_{s-1}^\top \mathbf{z}_s\right)^2 - \frac{1}{2} \sum_{s=1}^{t-1} \left(1 - \mathbf{u}^\top \mathbf{z}_s\right)^2 \leq d_0(\mathbf{u}, \mathbf{0}) - d_{t-1}(\mathbf{u}, \tilde{\mathbf{w}}_{t-1}) + 2 \sum_{s=1}^{t-1} h_s.$$

Isolating $d_{t-1}(\tilde{\mathbf{w}}_{t-1}, \mathbf{u})$, we get:

$$d_{t-1}(\mathbf{u}, \tilde{\mathbf{w}}_{t-1}) \leq d_0(\mathbf{u}, \mathbf{0}) + 2 \sum_{s=1}^{t-1} h_s - \frac{1}{2} \sum_{s=1}^{t-1} \left(1 - \tilde{\mathbf{w}}_{s-1}^\top \mathbf{z}_s\right)^2 + \frac{1}{2} \sum_{s=1}^{t-1} \left(1 - \mathbf{u}^\top \mathbf{z}_s\right)^2$$

Plugging this back into Eqn. (5) concludes the proof. \square

Lemma 4. *With the notation introduced so far, the following inequality holds for any (constant) $\mathbf{w} \in \mathbb{R}^D$, where $\mathbf{u} \in \mathbb{R}^D$ is the linear model approximating the reward function defined above:*

$$\mathbb{E} \left[\frac{1}{2} \left(1 - \mathbf{w}^\top \mathbf{z}_t\right)^2 - \frac{1}{2} \left(1 - \mathbf{u}^\top \mathbf{z}_t\right)^2 \right] \geq \frac{1}{2} (\hat{\sigma}_t - \sigma_t)^2 - 2q_t.$$

Proof. Let us define $\sigma_t = \frac{1}{2} (\Delta_{t,k_t} - \Delta_{t,l_t})$, and $\hat{\sigma}_t = \frac{1}{2} (\hat{\Delta}_{t,k_t} - \hat{\Delta}_{t,l_t})$. In addition, note that the probability of y_t being +1, marked p , is $\frac{1}{2} (1 + \frac{1}{2} (r(t, m_t) - r(t, n_t)))$. Based on the setting described above, observe the following:

$$\begin{aligned} \mathbb{E} \left\{ \left(1 - \mathbf{w}^\top \mathbf{z}_t\right)^2 - \left(1 - \mathbf{u}^\top \mathbf{z}_t\right)^2 \right\} \\ &= p \cdot [(1 - \hat{\sigma}_t)^2 - (1 - \sigma_t)^2] + (1 - p) \cdot [(1 + \hat{\sigma}_t)^2 - (1 + \sigma_t)^2] \\ &= p \cdot [(2 - \hat{\sigma}_t - \sigma_t) \cdot (\sigma_t - \hat{\sigma}_t)] + (1 - p) \cdot [(2 + \hat{\sigma}_t + \sigma_t) \cdot (\hat{\sigma}_t - \sigma_t)] \\ &= 4p \cdot (\sigma_t - \hat{\sigma}_t) + [(2 + \hat{\sigma}_t + \sigma_t) \cdot (\hat{\sigma}_t - \sigma_t)]. \end{aligned}$$

Now, since $|p - \frac{1+\sigma_t}{2}| \leq \frac{1}{2} q_t$, we have

$$\begin{aligned} \mathbb{E} \left\{ \left(1 - \mathbf{w}^\top \mathbf{z}_t\right)^2 - \left(1 - \mathbf{u}^\top \mathbf{z}_t\right)^2 \right\} &= 4p \cdot (\sigma_t - \hat{\sigma}_t) + [(2 + \hat{\sigma}_t + \sigma_t) \cdot (\hat{\sigma}_t - \sigma_t)] \\ &\geq 4 \left(\frac{1+\sigma_t}{2} - \frac{1}{2} q_t \right) \cdot (\sigma_t - \hat{\sigma}_t) + [(2 + \hat{\sigma}_t + \sigma_t) \cdot (\hat{\sigma}_t - \sigma_t)] \\ &= (\hat{\sigma}_t - \sigma_t)^2 - 2q_t \cdot (\sigma_t - \hat{\sigma}_t) \\ &\geq (\hat{\sigma}_t - \sigma_t)^2 - 2q_t \cdot |\sigma_t - \hat{\sigma}_t| \\ &\geq (\hat{\sigma}_t - \sigma_t)^2 - 4q_t. \end{aligned}$$

Adding the $\frac{1}{2}$ coefficient concludes the proof. \square

Lemma 5. *With the notation introduced so far, we have*

$$(\Delta_{t,m_t,n_t} - \hat{\Delta}_{t,m_t,n_t})^2 \leq 2\mathbf{z}_t^\top A_{t,k}^{-1} \mathbf{z}_t \cdot \left(d_0(\mathbf{u}, \mathbf{0}) + 2 \sum_{s=1}^t q_s + 2 \sum_{s=1}^{t-1} h_s + 36 \cdot \ln \frac{t+4}{\delta} \right),$$

with probability at least $1 - \delta$ uniformly over $t = 1, 2, \dots$.

Proof. The proof is identical to the proof of a similar lemma in [11], substituting \mathbf{x}_t for \mathbf{z}_t . \square

A.2 Proof of Theorem 1

Proof. We have with probability at least $1 - \delta$

$$R_T = \sum_{t=1}^T r_t \leq 2Q_T + 2 \sum_{t=1}^T \epsilon_t,$$

where $\epsilon_t^2 = 2\mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t \cdot \eta_t$. Using previous techniques, such as the one shown in [21], we have

$$\begin{aligned} \sum_{s=1}^{t-1} \mathbf{z}_s^\top A_s^{-1} \mathbf{z}_s &\leq \ln \frac{|A_{t-1}|}{|A_0|} \\ &= \ln |A_{t-1}| \\ &\leq dK \cdot \ln \left(1 + \frac{(t-1)}{dK} \right) \\ &\leq dK \cdot \ln \left(1 + \frac{T}{dK} \right). \end{aligned}$$

Observe that at round t , the matrix A_{t-1} is augmented by the PSD rank-one matrix $\mathbf{z}_t \mathbf{z}_t^\top$ regardless of the feedback. Hence, $\mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t \leq 1$. This means that $1 \leq \frac{2}{1 + \mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t}$, and thus,

$$\mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t \leq \frac{2 \cdot \mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t}{1 + \mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t} = 2\mathbf{z}_t^\top A_t^{-1} \mathbf{z}_t.$$

We can therefore combine the results above:

$$\begin{aligned}
\sum_{t=1}^T \epsilon_t^2 &= \sum_{t=1}^T 2\mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t \times \left(d_0(\mathbf{u}, \mathbf{0}) + 2 \sum_{s=1}^t q_t + 2 \sum_{s=1}^{t-1} h_s + 36 \ln \frac{t+4}{\delta} \right) \\
&\leq \left(d_0(\mathbf{u}, \mathbf{0}) + 2 \sum_{s=1}^t q_t + 2dK \ln \left(1 + \frac{T}{dK} \right) + 36 \ln \frac{T+4}{\delta} \right) \times 2 \sum_{t=1}^T \mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t \\
&\leq \left(d_0(\mathbf{u}, \mathbf{0}) + 2 \sum_{s=1}^t q_t + 2dK \ln \left(1 + \frac{T}{dK} \right) + 36 \ln \frac{T+4}{\delta} \right) \times 4 \sum_{t=1}^T \mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t \\
&\leq \left(d_0(\mathbf{u}, \mathbf{0}) + 2 \sum_{s=1}^t q_t + 2dK \ln \left(1 + \frac{T}{dK} \right) + 36 \ln \frac{T+4}{\delta} \right) \times 4dK \ln \left(1 + \frac{T}{dK} \right).
\end{aligned}$$

Observe that since $\sum_{t=1}^T \epsilon_t^2 \leq M$, it holds that $\sum_{t=1}^T \epsilon_t \leq \sqrt{T \cdot M}$. Combining this with all of the above, we get

$$\begin{aligned}
R_T &\leq 2Q_T + 2 \sum_{t=1}^T \epsilon_t \\
&\leq 2Q_T + \sqrt{d_0(\mathbf{u}, \mathbf{0}) + 2Q_T + 2dK \ln \left(1 + \frac{T}{dK} \right) + 36 \ln \frac{T+4}{\delta}} \times \sqrt{4TdK \ln \left(1 + \frac{T}{dK} \right)} \\
&= 2Q_T + \sqrt{2T} \left(\sqrt{((2Q_T + A)B)} + B \right),
\end{aligned}$$

thus proving the regret bound. \square

A.3 Equivalence to the Noisy Reward Setting

As mentioned before, we assume that the rewards $r(t, m)$ are as before, but feedback is given based on a noisy version of the reward, $\hat{r}(t, m)$, such that:

$$\hat{r}(t, m) = \begin{cases} +1 & w.p. \frac{1+r(t, m)}{2} \\ -1 & w.p. \frac{1-r(t, m)}{2} \end{cases}$$

meaning, the reward of an instance is shifted to either $+1$ or -1 with bias according to the unshifted reward. As for the feedback, we assume now that it is deterministic rather than stochastic, depending on the values of the noisy rewards of the two predictions, with one exception - if the rewards are identical (both $+1$ or -1), the feedback is ± 1 with equal probabilities. Let's now calculate the probability of receiving a $+1$ reward for predictions m_t, n_t :

$$\begin{aligned}
\Pr[y_t = +1] &= \Pr[\hat{\rho}(t, m_t) = +1 \wedge \hat{\rho}(t, m_t) = -1] \\
&\quad + \frac{1}{2} \cdot \Pr[\hat{\rho}(t, m_t) = +1 \wedge \hat{\rho}(t, m_t) = +1] \\
&\quad + \frac{1}{2} \cdot \Pr[\hat{\rho}(t, m_t) = -1 \wedge \hat{\rho}(t, m_t) = -1] \\
&= \left(\frac{1 + \rho(t, m_t)}{2} \right) \left(\frac{1 - \rho(t, n_t)}{2} \right) \\
&\quad + \frac{1}{2} \cdot \left(\frac{1 + \rho(t, m_t)}{2} \right) \left(\frac{1 + \rho(t, n_t)}{2} \right) \\
&\quad + \frac{1}{2} \cdot \left(\frac{1 - \rho(t, m_t)}{2} \right) \left(\frac{1 - \rho(t, n_t)}{2} \right) \\
&= \frac{1 + \frac{1}{2}(\rho(t, m_t) - \rho(t, n_t))}{2}
\end{aligned}$$

This is of course equivalent to the case where the rewards are clean and the feedback is stochastic we've shown before. Our regret now can either take into account the clean rewards as before, or the *expected* rewards, as follows:

$$r_t = \max_m \mathbb{E} [\hat{\rho}(t, m)] - \max \{ \mathbb{E} [\hat{\rho}(t, m_t)], \mathbb{E} [\hat{\rho}(t, n_t)] \}$$

Notice the following:

$$\begin{aligned} \mathbb{E} [\hat{\rho}(t, m)] &= (+1) \cdot \Pr [\hat{\rho}(t, m) = +1] + (-1) \cdot \Pr [\hat{\rho}(t, m) = -1] \\ &= \frac{1 + \rho(t, m)}{2} - \frac{1 - \rho(t, m)}{2} \\ &= \rho(t, m) \end{aligned}$$

Meaning the regret is also identical to before. This implies that the regret bound we've shown for the stochastic feedback setting applies here as well.

A.4 Dependency Parsing

In this setting, given a sentence $\mathbf{x}_t = (\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,n_t})$ in some natural language, we aim to find the best dependency parsing tree of that sentence. We assume a linear regret model, where the reward for a parsing \mathbf{m}_t for sentence \mathbf{x}_t is given by $\mathbf{u}^\top \Phi(\mathbf{x}_t, \mathbf{m}_t)$ for some $\mathbf{u} \in \mathbb{R}^D$ in the unit ball ($\|\mathbf{u}\| \leq 1$). This is done both for simplicity and since parse trees are often evaluated by the number of mistakes. As before, our algorithm maintains a weight vector $\mathbf{w}_t \in \mathbb{R}^D$, which is used to estimate the optimal dependency parsing. In addition, it maintain a PD matrix A_t , which is used to estimate our confidence in the relation between two possible parses. Finally, let $\tilde{\mathbf{w}}_t$ denote an orthogonal projection of \mathbf{w}_t that maintains $|\tilde{\mathbf{w}}_t^\top \Phi(\mathbf{x}_t, \mathbf{m})| \leq 1$ for any $\mathbf{m} \in \mathcal{K}_t$.

We find the optimal parsing by maximizing the following term:

$$\mathbf{m}_t = \arg \max_{\mathbf{m} \in \mathcal{K}_t} \tilde{\mathbf{w}}_{t-1}^\top \Phi(\mathbf{x}_t, \mathbf{m}).$$

In order to find the maximal \mathbf{m} , we construct from the sentence a full (directed) graph, where each word is a node, and the weight of an edge from word $\mathbf{x}_{t,i}$ to word $\mathbf{x}_{t,j}$ is given by $\tilde{\mathbf{w}}_{t-1}^\top \phi(\mathbf{x}_t, i, j)$. We can then use the Chu-Liu-Edmonds algorithm [22] to find the (non-projective) parsing \mathbf{m} which maximizes the term above, in $O(S^2)$ time.

Let $\pi_{\mathbf{m}}(j)$ denote the source of word $\mathbf{x}_{t,j}$ according to \mathbf{m} . After producing the believed-best parse tree, we select a word in the sentence (indexed by i_t) and a possible alternative source for it (indexed by j_t), such that our confidence in the source of \mathbf{x}_{t,i_t} being $\pi_{\mathbf{m}_t}(i_t)$ or \mathbf{x}_{t,j_t} is minimal, meaning that the confusion term is maximal. Algorithm 2 describes our solution for this setting.

For convenience, we define the following notations:

$$\begin{aligned} \epsilon_{t,i,j,k}^2 &= \eta_t \times (\phi(t, i, j) - \phi(t, k, j))^\top A_{t-1}^{-1} \\ &\quad \times (\phi(t, i, j) - \phi(t, k, j)) \\ \Delta_{t,i,j} &= \mathbf{u}^\top \phi(\mathbf{x}_t, i, j) \\ \Delta_{t,i,j,k} &= \frac{1}{2} (\Delta_{t,i,j} - \Delta_{t,k,j}) \\ \hat{\Delta}_{t,i,j} &= \tilde{\mathbf{w}}_{t-1}^\top \phi(\mathbf{x}_t, i, j) \\ \hat{\Delta}_{t,i,j,k} &= \frac{1}{2} (\hat{\Delta}_{t,i,j} - \hat{\Delta}_{t,k,j}) \end{aligned}$$

Regret Bound for the Dependency Parsing Setting

This setting is slightly different than the settings we have seen previously, but we can build on most of the results or proof techniques.

Lemma 6. *In the dependency parsing setting, if at time t the algorithm is such that $|\Delta_{t,i,j,k} - \hat{\Delta}_{t,i,j,k}| \leq \epsilon_{t,i,j,k}$ for all $i, j, k \in \{1, \dots, s_t\}$, then*

$$r_t = \max_{\mathbf{m} \in \mathcal{K}_t} \Delta_{t,\mathbf{m}} - \max \{ \Delta_{t,\mathbf{m}_t}, \Delta_{t,\mathbf{n}_t} \} \leq 2S\epsilon_t$$

where

$$\epsilon_t^2 = 2\mathbf{z}_t^\top A_{t-1}^{-1} \mathbf{z}_t \times \eta_t$$

for all $t = 1, 2, \dots$

Algorithm 2 CONQUER Dependency Parsing - Algorithm Outline

Input: $\delta \in (0, 1)$

```

1: Initialize  $\mathbf{w}_0 = \mathbf{0} \in \mathbb{R}^D$ ,  $A_0 = \mathbf{I}_{D \times D}$ 
2: for  $t = 1$  to  $T$  do
3:   Receive sentence  $\mathbf{x}_t = (\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,s_t})$ 
4:   Project  $\tilde{\mathbf{w}}_{t-1} = \arg \min d_t(\mathbf{w}, \mathbf{w}_{t-1})$ 
   Subject To  $|\mathbf{w}^\top \phi(\mathbf{x}_t, i, j)| \leq 1 \ \forall i, j \in \{1, \dots, s_t\}, i \neq j$ 
5:   Set  $\mathbf{m}_t = \arg \max_{\mathbf{m} \in \mathcal{K}_t} \hat{\Delta}_{t,\mathbf{m}}$ 
6:   for all  $(k, j) \in \mathbf{m}_t$  do
7:     for all  $i \in 1, \dots, s_t, i \neq k, j$  do
8:       Set  $\beta(i, j) = \epsilon_{t,k,j,i}$ 
9:     end for
10:  end for
11:  Set  $(i_t, j_t) = \arg \max_{i,j} \beta(i, j)$ 
12:  Set  $\mathbf{n}_t$  such that  $\pi_{\mathbf{n}_t}(j) = \begin{cases} \pi_{\mathbf{m}_t}(j) & \text{if } j \neq j_t \\ i_t & \text{if } j = j_t \end{cases}$ 
13:  Output  $\mathbf{m}_t, \mathbf{n}_t$ 
14:  Receive feedback  $y_t \in \{\pm 1\}$ 
15:  Set  $\mathbf{z}_t = \frac{1}{2} y_t \cdot (\Phi(\mathbf{x}_t, \mathbf{m}_t) - \Phi(\mathbf{x}_t, \mathbf{n}_t))$ 
16:  Update  $\mathbf{w}_t = A_t^{-1} (A_{t-1} \tilde{\mathbf{w}}_{t-1} + \mathbf{z}_t)$ 
17:  Update  $A_t = A_{t-1} + \mathbf{z}_t \mathbf{z}_t^\top$ 
18: end for
Output:  $\tilde{\mathbf{w}}_T$ 

```

Proof. Similarly to Lemma 2, we have:

$$\begin{aligned}
r_t &= \Delta_t^* - \max \{ \Delta_{t,\mathbf{m}_t}, \Delta_{t,\mathbf{n}_t} \} \\
&\leq \Delta_t^* - \Delta_{t,\mathbf{m}_t} \\
&= 2 \sum_{i=1}^{s_t} \Delta_{t,\pi_{\mathbf{m}_t^*}(i),i,\pi_{\mathbf{m}_t}(i)} \\
&\leq 2 \sum_{i=1}^{s_t} \hat{\Delta}_{t,\pi_{\mathbf{m}_t^*}(i),i,\pi_{\mathbf{m}_t}(i)} + \epsilon_{t,\pi_{\mathbf{m}_t^*}(i),i,\pi_{\mathbf{m}_t}(i)} \\
&= \hat{\Delta}_t^* - \hat{\Delta}_{t,\mathbf{m}_t} + 2 \sum_{i=1}^{s_t} \epsilon_{t,\pi_{\mathbf{m}_t^*}(i),i,\pi_{\mathbf{m}_t}(i)} \\
&\leq 2S \cdot \epsilon_{t,i_t,j_t,\pi_{\mathbf{m}_t}(j_t)} \\
&= 2S \times \epsilon_t.
\end{aligned}$$

□

Theorem 7. In the dependency parsing setting described so far, the cumulative regret R_t of the algorithm satisfies

$$R_T = O \left(S \times \sqrt{2T} \left(\sqrt{AB} + B \right) \right),$$

where

$$\begin{aligned}
A &= d_0(\mathbf{u}, \mathbf{0}) + 36 \cdot \ln \frac{T+4}{\delta} \\
B &= 2dK \ln \left(1 + \frac{T}{dK} \right),
\end{aligned}$$

with probability at least $1 - \delta$ uniformly over the time horizon T .

Proof. From Lemma 6, we have the following result:

$$R_T = \sum_{t=1}^T r_t \leq 2S \sum_{t=1}^T \epsilon_t.$$

We can now use the exact same path as in Theorem 1 to show the regret bound.

□

	CNQR-GNC (ours)	Confidit [11]	Partial Feedback [16]
Parameters	None	$\alpha \in (-1, 1]$	Constant $a \in [0, 1]$ Cost values $c(i, s)$ Interval $D \in [-R, R]$ Function $g : D \rightarrow \mathcal{R}$
Model	$\mathbf{w}_t \in \mathbb{R}^D$ $A_t \in \mathbb{R}^{D \times D}$		$\mathbf{w}_{i,t} \in \mathbb{R}^d, i = 1, \dots, K$ $A_{i,t} \in \mathbb{R}^{d \times d}, i = 1, \dots, K$
Predictions	$m_t = \arg \max_m \hat{\Delta}_{t,m}$ $n_t = \arg \max_n \hat{\Delta}_{t,n} + \epsilon_{t,m_t,n}$	$m_t = \arg \max_m \hat{\Delta}_{t,m} + \epsilon_{t,m}$	$\hat{Y}_t = \arg \max_Y$ $(1-a) \sum_{i \in Y} (c(j_i, Y) - \left(\frac{a}{1-a} + c(j_i, Y)\right) \hat{p}_{i,t})$
Number of Predictions	2	1	$1 \leq \hat{Y}_t \leq K$
Feedback Type	Stochastic (Bernoulli)	Deterministic	
Feedback	$\Pr[y_t = \pm 1]$ $= \frac{1}{2} \left(1 \pm \frac{1}{2} (r(t, m_t) - r(t, n_t))\right)$	$y_t = \begin{cases} 1 & m_t^* \neq m_t \\ 0 & m_t^* = m_t \end{cases}$	$Y_t \cap \hat{Y}_t$
Update	$\mathbf{z}_t = \frac{1}{2} (\phi(\mathbf{x}_t, m_t) - \phi(\mathbf{x}_t, n_t))$ $A_t = A_{t-1} + \mathbf{z}_t \mathbf{z}_t^\top$ $\mathbf{w}_t = \tilde{\mathbf{w}}_{t-1} + (y_t - \hat{\Delta}_{t,m_t,n_t}) A_t^{-1} \mathbf{z}_t$	$s_t = \begin{cases} +1 & y_t = 0 \\ +1 & y_t = 1, \text{w.p. } \frac{1-\alpha}{2} \\ -1 & y_t = 1, \text{w.p. } \frac{1+\alpha}{2} \end{cases}$ $\mathbf{z}_t = \phi(\mathbf{x}_t, m_t)$ $A_t = A_{t-1} + \mathbf{z}_t \mathbf{z}_t^\top$ $\mathbf{w}_t = \tilde{\mathbf{w}}_{t-1} + (s_t - \hat{\Delta}_{t,m_t}) A_t^{-1} \mathbf{z}_t$	$s_{i,t} = \begin{cases} +1 & i \in Y_t \cap \hat{Y}_t \\ -1 & i \in \hat{Y}_t / Y_t \\ 0 & \text{otherwise} \end{cases}$ $\mathbf{z}_t = \mathbf{x}_t$ $A_{i,t} = A_{i,t-1} + \mathbf{z}_t \mathbf{z}_t^\top$ $\mathbf{w}_{i,t} = \tilde{\mathbf{w}}_{i,t} + \frac{g(s_{i,t}, \hat{\Delta}_{t,i,t}) s_{i,t}}{c_{i,t}} A_{i,t}^{-1} \mathbf{z}_t$
Output	\mathbf{w}_T		

Table 1: A summary of contextual bandits methods with partial feedback

A.5 Computing the Projection

Denote by \mathbf{v}^i the vector computed after the i th iteration. Then, the new vector at this point is given by, $\mathbf{v}^{i+1} = \arg \min_{\mathbf{w}} d_t(\mathbf{w}, \mathbf{v}^i)$ s.t. $|\mathbf{w}^\top \Phi(\mathbf{x}_t, m(i))| \leq 1$, for some $m(i) \in \mathcal{K}$, which can be easily solved using Lagrange's method. The solution is given by,

$$\mathbf{v}^{i+1} = \mathbf{v}^i - A_t^{-1} \Phi(\mathbf{x}_t, m(i)) \cdot \text{sign}(\mathbf{v}^i \cdot \Phi(\mathbf{x}_t, m(i))) \cdot \frac{\max\{|\mathbf{v}^i \cdot \Phi(\mathbf{x}_t, m(i))| - 1, 0\}}{\Phi(\mathbf{x}_t, m(i))^\top A_t^{-1} \Phi(\mathbf{x}_t, m(i))}. \quad (6)$$

To conclude, the projection algorithm sets $\mathbf{v}^1 = \mathbf{w}_{t-1}$. It then iterates. On the i th iteration, it picks a (random) index $m = m(i)$ and sets \mathbf{v}^{i+1} according to the three cases of \mathbf{v}^i sketched above. This process is repeated until some convergence criteria are met, and then the last value vector \mathbf{v}^i is returned. The time complexity is $O(D^2 + nD^2)$, where D^2 is the time needed to compute the inverse A_t^{-1} if performed incrementally, D^2 the time needed to compute each iteration, and n the number of iterations performed.

A.6 Additional Related Work

Table 2 and Table 1 provide some context for our setting compared with other similar settings, enabling a side-by-side comparison and a better understanding of our solution compared with existing works.

A.7 Additional Results

We include additional results from our study, which could not be fitted into the paper itself. Fig. 5 shows the test errors of the four tested algorithms, over an additional 9 domains. These results correspond to the average and aggregated results we included in the paper. Finally, Fig. 6 demonstrates the performance of the tested algorithms in terms of online regret over 3 domains. It is evident that good performance in terms of test error corresponds with good performance in terms of online loss, which is an indication that online loss is a valid estimate of algorithm performance in this setting.

Finally, Table 3 shows in what percent of rounds CNQR-GNC avoided making a query on average, per domain and number of reviews per round.

Algorithm Name	Regret Formulation	Feedback Type	Assumptions	Algorithm Type	Context
<i>Interleaved Filter</i> [1]	<p>Strong regret: $\Pr \{b_{m_t^*} > b_{m_t}\} + \Pr \{b_{m_t^*} > b_{n_t}\}$</p> <p>Weak regret: $\min \left[\Pr \{b_{m_t^*} > b_{m_t}\}, \Pr \{b_{m_t^*} > b_{n_t}\} \right]$</p>	Relative Stochastic	<ul style="list-style-type: none"> - Stochastic Triangle Inequality - Strong Stochastic Transitivity 	Survival	No
<i>Beat the Mean</i> [2]	$\frac{1}{2} \Pr \{b_{m_t^*} > b_{m_t}\} + \frac{1}{2} \Pr \{b_{m_t^*} > b_{n_t}\}$	Relative Stochastic	<ul style="list-style-type: none"> - Stochastic Triangle Inequality - Relaxed Stochastic Transitivity 	Survival	No
<i>Confidit</i> [11]	$\max_m [\Pr (m_t^* = m)] - \Pr (m_t^* = m_t)$	Absolute Deterministic	Generative model for labels	UCB Second Order	Yes
<i>Partial Feedback</i> [16]	$(1-a) \sum_{i \in \hat{Y}_t} \left(c(j_i, \hat{Y}_t) \right) - \left(\frac{a}{1-a} + c(j_i, \hat{Y}_t) \right) \mathbb{1}_{i \in Y_t}$	Absolute Deterministic	Specific form for marginals	Linear Second Order	Yes
<i>CNQR-GNC</i> (ours)	$\min [r(\mathbf{x}_t, m_t^*) - r(\mathbf{x}_t, m_t), r(\mathbf{x}_t, m_t^*) - r(\mathbf{x}_t, n_t)]$	Relative Stochastic linear model	Reward can be approximated using Second Order	Mixed (UCB & Linear)	Yes

Table 2: A comparison of bandits methods with partial feedback

Number of Reviews	5	10	15	20
<i>Android</i>	0.53%	0.79%	0.53%	0.26%
<i>Arts</i>	0.26%	0.00%	0.00%	0.00%
<i>Automotive</i>	0.64%	0.21%	0.00%	0.00%
<i>Baby Products</i>	1.73%	0.11%	0.05%	0.05%
<i>Beauty</i>	2.00%	0.16%	0.11%	0.05%
<i>Books</i>	0.07%	0.01%	0.01%	0.00%
<i>Camera</i>	0.59%	0.29%	0.03%	0.08%
<i>Cell Phones</i>	1.17%	0.11%	0.05%	0.03%
<i>Clothing</i>	4.56%	0.83%	0.32%	0.16%
<i>Computers</i>	0.27%	0.00%	0.05%	0.00%
<i>Electronics</i>	0.76%	0.19%	0.13%	0.07%
<i>Food</i>	1.41%	0.37%	0.11%	0.13%
<i>Garden & Pets</i>	1.23%	0.16%	0.05%	0.00%
<i>Health</i>	2.01%	0.41%	0.12%	0.08%
<i>Home</i>	0.59%	0.21%	0.03%	0.00%
<i>Industrial</i>	0.00%	0.00%	0.00%	0.00%
<i>Jewelry</i>	0.53%	0.26%	0.26%	0.00%
<i>Kindle</i>	0.24%	0.05%	0.00%	0.01%
<i>Kitchen</i>	1.96%	0.43%	0.08%	0.01%
<i>Magazines</i>	0.00%	0.00%	0.00%	0.00%
<i>Movies & TV</i>	0.48%	0.21%	0.12%	0.01%
<i>MP3</i>	0.40%	0.12%	0.04%	0.11%
<i>Music</i>	1.21%	0.59%	0.41%	0.12%
<i>Musical Instruments</i>	0.00%	0.00%	0.00%	0.00%
<i>Office</i>	1.01%	0.16%	0.16%	0.00%
<i>Patio</i>	0.80%	0.19%	0.08%	0.00%
<i>Shoes</i>	8.27%	1.44%	0.75%	0.51%
<i>Software</i>	0.48%	0.05%	0.03%	0.00%
<i>Sports</i>	0.51%	0.19%	0.11%	0.05%
<i>Toys</i>	1.65%	0.43%	0.69%	0.43%
<i>Video Games</i>	0.48%	0.00%	0.16%	0.05%
<i>Videos</i>	0.12%	0.01%	0.00%	0.01%
<i>Watches</i>	0.00%	0.00%	0.00%	0.00%

Table 3: Percent of rounds where query was not made in CNQR-GNC for each domain

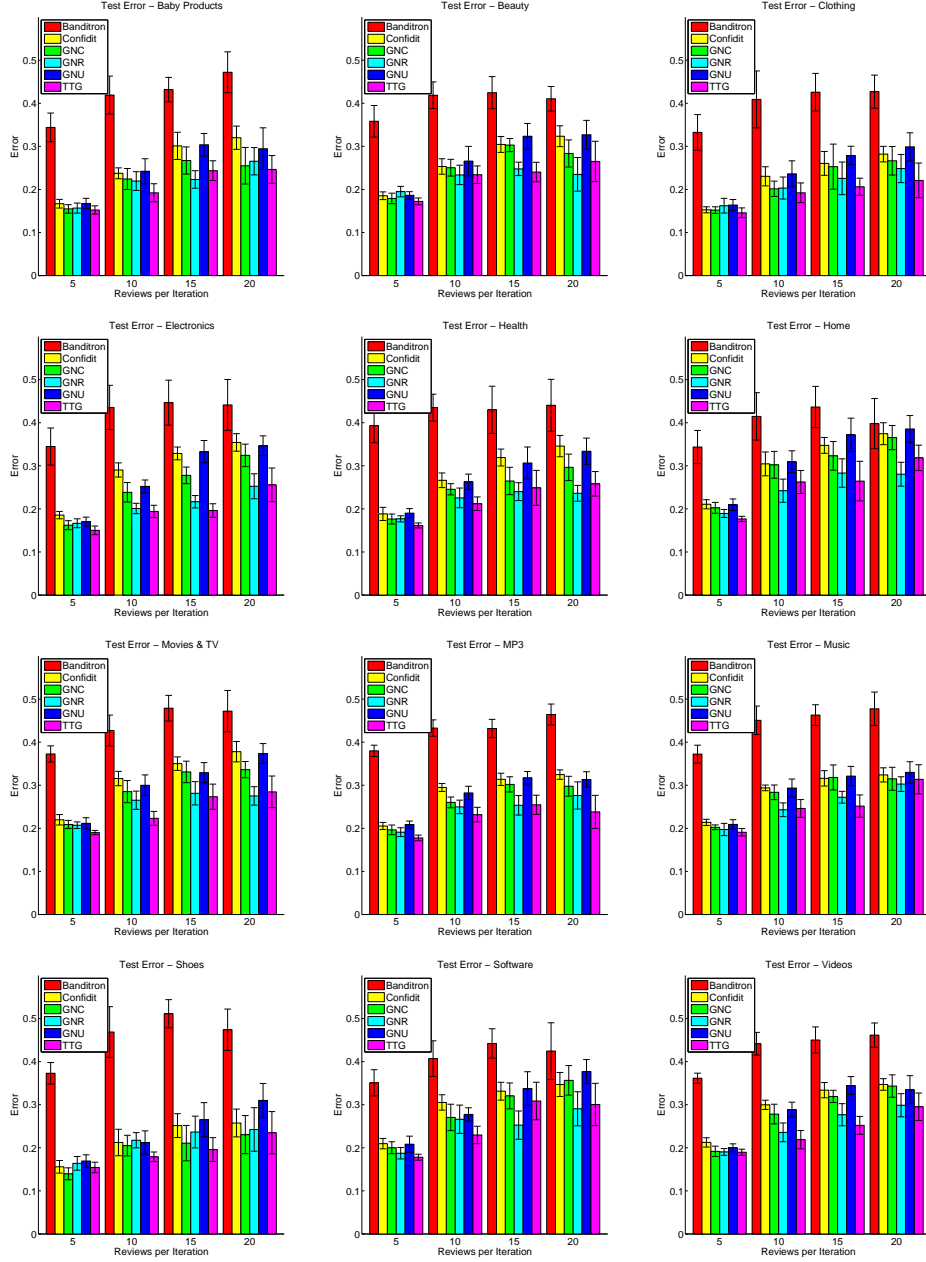


Figure 5: Average test error over Baby Products, Beauty, Clothing, Electronics, Health, Home, Movies & TV, MP3, Music, Shoes, Software and Videos domains.

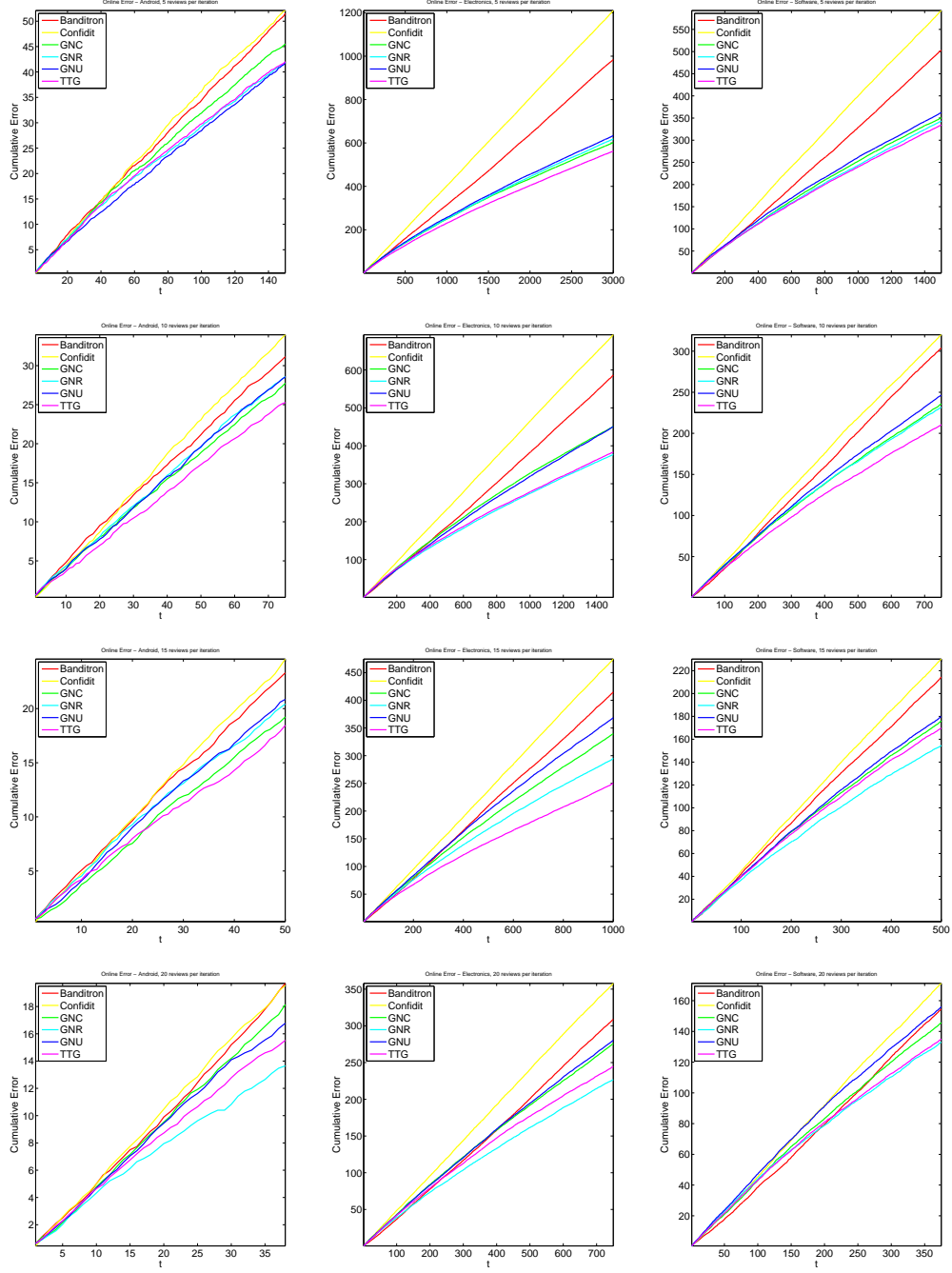


Figure 6: Cumulative online regret over Android, Electronics and Software domains.